REMARKS

Applicants respectfully request reconsideration and allowance of the present application. By this Amendment, Applicants amend claim 1, 9, 11, 13, 26, 32, 35 and 43, cancel claim 52 and add claim 54. Claims 1-50 and 53-54 will be pending in the application upon entry of this Amendment.

Claim Rejections Under 35 U.S.C. § 102

Claims 1-4 and 8-18 stand rejected under 35 USC § 102(e) as being taught by U.S. Patent No. 5,989,865 to Mahalingaiah ("Mahalingaiah"). For reasons set forth more fully below, this rejection is respectfully traversed.

Mahalingaiah Does Not Store Instructions Received From A Fetch Stage As Required By Amended Independent Claim 1

Amended independent claim 1 requires, *inter alia*, a buffer in a dispatch stage that is adapted to **receive from a fetch stage** and store a set of loop instructions and scheduling information associated with the instructions. The instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the loop body.

The Office Action points to Mahalingaiah's MROM Access as corresponding to the claimed buffer. However, MROM Access merely stores microcode corresponding to MROM instructions. MROM Access is essentially a fixed lookup table (e.g. ROM) that contains the set of microcode instructions corresponding to MROM instructions. Mahalingaiah does not describe how MROM Access is configured, much less disclose or suggest that MROM Access ever stores instructions that are received from a fetch stage, as is required by amended independent claim 1. Presumably the ROM is configured in an off-line process.

For at least these reasons, amended independent claim 1 patentably defines over Mahalingaiah and the § 102 rejection of claim 1, together with claims 2-4 and 8-18 that depend therefrom, should be withdrawn.

Claim 4 Further Patentably Defines Over Mahalingaiah

Claim 4 requires, with subject matter from claim 3:

> control logic coupled to the buffer adapted to cause a certain one of the stored plurality of instructions to be issued to the functional unit in accordance with a loop iteration stage and the stored scheduling information associated with the certain instruction
>
> . . . .
>
> wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with a cycle within the loop iteration stage and the stored scheduling information associated with the certain instruction.

The Office Action, in responding to Applicants' evidence that this subject matter is not taught or suggested by Mahalingaiah, takes the position that :

> The claim language in question, can mean a cycle when the loop is still executing, e.g. where the number of loop iterations is still being tracked. Applicants' arguments seem to be arguing a narrower definition of the phrase "a cycle within the loop iteration stage" than is explicitly defined in the specification and claim. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., a cycle within the loop iteration stage) are not recited in the rejected claim(s).

Applicants respectfully submit that this position in the Office Action ignores clear and explicit claim limitations and takes positions that are contrary to those skilled in the art and Mahalingaiah's own explicit teachings.

The claim explicitly requires causing "**the certain one of the instructions to be issued in accordance with a cycle within the loop iteration stage and the stored scheduling information associated with the certain instruction**." Applicants agree with the Office Action that a "cycle within the loop iteration stage" can mean a cycle "when the loop is still executing." However, Applicants respectfully disagree with the Office Action to the extent it suggests that the claim merely requires that "instructions are issued from the buffer during cycles whenever the loop is still executing." The claim requires much more, and certainly much more than Mahalingaiah discloses or suggests.

Even with the Office Action's interpretation, Mahalingaiah does not cause a **certain** instruction **to be issued** in accordance with a cycle "when the loop is still executing." Mahalingaiah merely **stops** issuing instructions from MROM access based on when a number of **iterations** of a string instruction has completed. It does not teach or suggest causing "the **certain one** of the instructions **to be issued** in accordance with **a cycle within the loop iteration stage**" as explicitly required by the claim.

Mahalingaiah only teaches generating an address within MROM Access for a line of **multiple** instructions to be taken from MROM Access based on whether a loop branch is predicted to be taken. It does not generate addresses for a certain **one** of the instructions, much less in accordance with a cycle within a given loop iteration stage "when the loop is still executing." Mahalingaiah merely teaches at col. 17, lines 58-65: "After each iteration of the loop, the value in ECX is decremented. If the value of ECX, after being decremented, is non-zero, the microcode sequence branches to the beginning of the loop and repeats the microcode loop instructions. After the loop has been implemented the number of times specified by the ECX register, the loop is terminated and the exit microcode instruction is implemented."

For at least the foregoing reasons, claim 4 further patentably defines over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claims 8, 10, 12 and 14 Further Patentably Defines Over Mahalingaiah

Claim 8 requires, with similar subject matter in claims 10, 12 an 14:

> control logic coupled to the buffer, the control logic including:
> an iteration initiation interval register for storing a loop iteration initiation parameter;
> a loop iteration register for storing a loop iteration parameter; and
> a loop cycles register for storing a loop cycles parameter,
> wherein the control logic is adapted to cause the plurality of instructions to be issued to the functional unit from the buffer in accordance with the loop iteration initiation parameter, the loop iteration parameter, the loop cycles parameter and the stored scheduling information.

The Office Action, in responding to Applicants' evidence that this subject matter is not taught or suggested by Mahalingaiah, takes the position that :

> Mahilingaiah needs three pieces of information to determine if the loop can be exited or cancelled: the string count, number of iterations of the microcode loop, and whether the two are greater than or equal to each other. These are the loop iteration initiation parameter, loop iteration parameter, and the loop cycles parameter. The initiation parameter is the string count, since a string that requires a loop starts a loop. The loop iteration parameter is the number of iterations the microcode loop has executed, since that is the loop iteration currently being executed. The loop cycles parameter is the comparison result, since it represents whether the loop cycle is complete or not. Applicants' arguments seem to be suggesting a meaning to these terms that is not reflected in the claim language nor is there an explicit definition within the specification. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., the loop iteration initiation parameter, loop iteration parameter, and the loop cycles parameter) are not recited in the rejected claim(s).

Applicants respectfully submit that this position in the Office Action ignores clear and explicit claim limitations and takes positions that are self-contradictory and contrary to those skilled in the art and Mahalingaiah's own explicit teachings.

First, it is believed to be notoriously well known by those skilled in the art what a "loop iteration" and a "loop cycle" is, so explicit definitions of these notoriously well known terms is not considered necessary. Indeed, Mahalingaiah itself clearly distinguishes between "iterations" and "cycles" in accordance with these well-known terms. (compare, for example, col. 17, line 8-14 with col. 17, line 51-59). Moreover, the present specification is replete with references that are consistent with these notoriously well-known terms (see e.g. page 12, line 11 to page 13, line 4).

Meanwhile, the Office Action states that "The loop iteration parameter is the number of iterations the microcode loop has executed, since that is the loop iteration currently being executed. The loop cycles parameter is the comparison result, since it represents whether the loop cycle is complete or not." Applicants agree that a loop iteration can be a "number of iterations" of the loop. However, Applicants respectfully disagree that a loop cycle can be a result indicating "whether the loop cycle is complete or not." Apart from being an unreasonable interpretation by one skilled in the art, it is self-contradictory and inconsistent with Mahalingaiah.

Mahalingaiah merely discusses tracking the number of loop iterations to determine a branch condition. The Office Action impermissibly attempts to stretch Mahalingaiah's tracking of loop iterations to cover three different claimed quantities.

Specifically, to support its rejection, the Office Action's interpretation seems to imply that "loop iteration" is met by Mahalingaiah's current loop iteration and the "loop cycle" is met by Mahalingiah's "comparison result" of whether the loop iteration is the last one. In addition to flying in the face of well-understood meaning of "loop cycles," this rationale merely tries to split the same information into two different components. Moreover, the Office Action also says that the "loop iteration initiation" is met by Mahalingaih's string count, which is just the total number of loop iterations. So the Office Action effectively attempts to stretch Mahalingaiah's tracking of loop iterations (i.e. string count) to cover three different claim elements.

Rather than interpreting the claims "broadly," the Office Action is impermissibly ignoring clear and explicit claim limitations. Under the Office Action's interpretation, the three different parameters could all refer to the same quantity, which does not respect the claim language, in addition to being offensive to the understanding of those skilled in the art.

For at least the foregoing reasons, claims 8, 10, 12 and 14 further patentably define over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claims 9, 11 and 13 Further Patentably Define Over Mahalingaiah

Claims 9, 11 and 13 depend directly or indirectly from patentable claim 1 and are patentable for at least the reasons claim 1 is patentable.

Claims 9, 11 and 13 have been amended to require that the stored instructions **consist of** a set of kernel instructions and to further require control logic in the processor that is "operative so that the functional unit executes a number of loop iterations of the stored kernel set of loop instructions, a prologue set of loop instructions different than the **stored** kernel set of loop instructions, and an epilogue set of loop instructions different than the **stored** kernel set of loop instructions based on the **stored** kernel set of loop instructions and received loop parameters."

Mahalingaiah discloses nothing about prologue and epilogue sets of loop instructions that are **different than** a kernel set of loop instructions, much less control logic that is operative to such that the functional unit executes the number of iterations of the kernel set of loop

instructions, the **different** prologue set of loop instructions and the epilogue set of loop instructions **based on the stored kernel loop instructions** and the received loop parameters.

Moreover, the Office Action, without providing support from Mahalingaiah, states that "Mahalingaiah has taught that the instructions are all stored in the same location. This includes the kernel, prologue, and epilogue instructions." Accordingly, by the Office Action's admission, Mahalingaiah would only suggest requiring that all sets of instructions that are executed must also be stored, contrary to the clear limitations of the claim.

For at least the foregoing reasons, claims 9, 11 and 13 further patentably define over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claims 15-18 Further Patentably Defines Over Mahalingaiah

Claims 15 and 16 ultimately depend from patentable claim 1 and further require that the control logic is operative so that the **fetch unit can be shut down** during execution of a number of iterations of the loop body corresponding to the stored plurality of instructions. Claims 17 and 18 also require that the control logic is operative so that the fetch unit can be shut down during loops.

The Office Action states that "[s]hutting down a fetch unit merely means that instructions are no longer fetched. When a fetch unit is stalled, the instructions are no longer fetched." This argument, even if valid, **fails** because of the irrefutable fact that **Mahalangiah does not even disclose ever stalling a fetch unit.** Mahalingaiah merely states that "Fastpath instructions from **instruction alignment unit 18** are stalled while MROM microcode instructions are issued by MROM unit 34." This merely states that the conveyance of instructions from unit 18 to the functional units is delayed (i.e. the unit is not shut down) while MROM instructions (e.g. string instructions) are issued from MROM unit 34. Mahalingaiah's fetch unit 70 is totally separate from alignment unit 18. Moreover, Mahilingaiah fails to disclose or suggest that fetch unit 70 is **ever** capable of being shut down, much less during MROM instruction execution.

For at least the foregoing reasons, claims 15-18 further patentably define over Mahalingaiah and the § 102 rejection thereof should be withdrawn.

Claim Rejections Under 35 U.S.C. § 103 In View Of Mahalingaiah and Subramanian

Claims 5-7, 26-33, 35-41 and 43-49 stand rejected under 35 USC 103(a) as being unpatentable over Mahalingaiah in view of U.S. Patent No. 5,867,711 to Subramanian et al. ("Subramanian").[1] For at least the foregoing reasons, this rejection is respectfully traversed.

Claims 5-7 Patentably Define Over Mahalingaiah and Subramanian

Claims 5-7 depend from claim 1 and are patentable for at least the reasons claim 1 is patentable. Claims 5-7 further require:

> 1. . . . a buffer in the dispatch stage coupled to the functional unit adapted to receive from a fetch stage and store a plurality of the instructions before issue to the functional unit and further adapted to store scheduling information associated with the instructions . . . wherein the instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the loop body.
>
> 5. A processor according to claim 1, wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions.
>
> 6. A processor according to claim 3,
> wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions, and
> wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with the respective one of the loop stage bit masks associated with the certain one of the instructions.
>
> 7. A processor according to claim 4,
> wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions, and
> wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with the respective one of the loop stage bit masks associated with the certain one of the instructions.

---

[1] The Office Action did not list claim 53 together with these other claims, but addressed claim 53 in paragraph 23 along with claim 32, and also stated in paragraph 27 that claim 53 was rejected for the same reasons set forth for claim 32. Moreover, the Office Action in paragraph 40 rejected "Claim 53," but the reasoning appeared to be referring to contents of claim 52, which has been canceled herein.

The Office Action relies on Subramanian for meeting this subject matter that is admittedly missing from Mahalingaiah. However, the Office Action utterly fails to establish a **prima facie** case of obviousness, because all claim limitations are not disclosed or suggested by Subramanian and Mahalingaiah, even if they could be combined as alleged.

Subramanian merely teaches a compiler that analyzes a target program instruction loop, generates a data dependency graph, and creates a modulo schedule from that graph. The compiler then represents the schedule in an instruction sequence that consists of a Prolog, Kernel and Epilog (PKE form). In Subramanian's teaching, the prolog and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target processor hardware presumably merely executes that code in that form.

The Office Action takes the position that the following passage from Subramainian describes the claimed loop stage bit masks:

> A procedure is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be understood, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Meanwhile, claim 5 (with subject matter incorporated from claim 1) clearly requires: "a buffer in the dispatch stage coupled to the functional unit adapted to receive from a fetch stage and store a plurality of the instructions before issue to the functional unit and further adapted to store scheduling information associated with the instructions . . . wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions. . . [and] wherein the instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the loop body."

Subramanian has been searched exhaustively and there are no "bit masks" taught or suggested at all, much less loop stage bit masks respectively associated with a plurality of instructions, which are then used to selectively issue the instructions to a functional unit as required by claim 5.

For at least these reasons, the § 103 rejections of claims 5-7 should be withdrawn.

Claims 26-31 Patentably Define Over Mahalingaiah and Subramanian

Independent claim 26, from which claims 27-31 depend, has been amended to more clearly require that instructions stored in the buffer are **received from a fetch stage**. Accordingly, claim 26 is patentable for at least the reasons claim 1 is patentable as set forth above.

Similar to claims 5-7, independent claim 26 also requires storing modulo schedule stage identifiers respectively associated with the stored plurality of instructions, which are used to cause associated instructions to be selectively issued to a functional unit. The Office Action relies on Subramanian for meeting this subject matter admittedly missing from Mahalingaiah.

As set forth above, Subramanian only teaches compiler software for generating a modulo-scheduled program from a dependency graph generated from a target program instruction loop. As part of this process, Subramanian maps physical times for instructions in generated graph so as to arrive at a schedule for instructions in the output program. (see col. 10, lines 5-9).

The Office Action states that "Whatever can be executed in software can be implemented in hardware." However, at best, this argument supports the notion that one skilled in the art using Subramanian's disclosure could build hardware that generates an output program from a dependency graph by mapping instructions in the graph according to associated physical times. It would not suggest hardware that stores modulo schedule stage identifiers associated with instructions that are used to cause the instructions to be selectively issued to a functional unit, as explicitly required claim 26.

For at least these reasons, the § 103 rejection of independent claim 26, together with claims 27-31 that depend therefrom, should be withdrawn.

Amended Independent Claim 32 Patentably Defines Over Mahalingaiah and Subramanian

Independent claim 32 has been amended to clearly require buffers in the dispatch stage "respectively associated with" a plurality of functional units for storing a kernel set of loop instructions and a different set of scheduling information associated with the respective functional unit, and:

> control logic coupled to the plurality of buffers for causing the stored kernel set of instructions to be selectively issued to the respective functional units in accordance with the stored scheduling information, the control logic being operative so that the functional units execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored kernel set of loop instructions and associated scheduling information, **wherein during a first processor cycle, the control logic uses the sets of scheduling information to cause a portion of the prologue set of loop instructions to be issued to the respective functional units from the buffers, and wherein during a second processor cycle after the first processor cycle, the control logic uses the sets of scheduling information to cause a portion of the kernel set of loop instructions to be issued to the respective functional units from the buffers, and wherein during a third processor cycle after the second processor cycle, the control logic uses the sets of scheduling information to cause a portion of the epilogue set of loop instructions to be issued to the respective functional units from the buffers**.

Mahalingaiah's MROM Access contains "lines" of instructions for execution by all the functional units. Mahalingaiah does not disclose or suggest the control logic's use of the scheduling information to cause issuance of prologue, kernel and epilogue sets of instructions at various processor cycles as explicitly required by amended claim 32, much less providing respective buffers for each functional unit.

Moreover, as set forth more fully above, Subramanian teaches compiler software methods for creating a modulo schedule from a target program loop, and does not suggest control logic in a processor. Nothing in Subramanian discloses or suggests the operation of different functional units during loop execution, much less storing respective sets of different scheduling information for each as required by claim 32.

For at least the foregoing reasons, claim 32, together with claims 33 and 34 that depend therefrom, patentably define over Mahalingaiah and Subramanian and the § 103 rejection thereof should be withdrawn.

Independent Claim 53 Patentably Defines Over Mahalingaiah and Subramanian

The Office Action stated that claim 53 was rejected "for the same reasons" as claim 32. However, claim 53 has many limitations that do not appear in claim 32. For example, claim 53 explicitly requires:

> control logic coupled to the buffer, the control logic being adapted to:
> receive loop parameters associated with **loop instructions issued to the functional unit from a fetch stage**;
> **cause a kernel set of the loop instructions issued to the functional unit to be stored in the buffer** in accordance with the received loop parameters. . . .

For similar reasons as set forth above in connection with claim 1, Mahalingaiah's MROM Access does not cause **instructions that are issued from a fetch stage** to be stored in the buffer, which is fixed at run-time.

Moreover, as set forth more fully above, Subramanian merely teaches compiler software methods for creating a modulo schedule from a target program loop, and does not suggest control logic in a processor.

For at least the foregoing reasons, claim 53 patentably defines over Mahalingaiah and Subramanian and the § 103 rejection thereof should be withdrawn.

Amended Independent Claims 35 and 43 Patentably Define Over Mahalingaiah and Subramanian

Independent claims 35 and 43 have been amended to require that instructions stored in the buffer are **received from a fetch stage**. Accordingly, claims 35 and 43 are patentable for at least the reasons claim 1 is patentable as set forth above. Moreover, claims 35 and 43 require:

- Loop parameters are stored "in control logic of the processor";
- The stored kernel set of loop instructions are caused "to be selectively issued to functional units of the processor" in accordance with the stored loop parameters; and

- The selective issuance of the kernel set of instructions is done "so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions."

Mahilingaiah admittedly teaches nothing about prologue, kernel and epilogue sets of loop instructions.

As set forth above, Subramanian only teaches compiler software for generating a modulo-scheduled program from a dependency graph generated from a target program instruction loop. As part of this process, Subramanian maps physical times for instructions in generated graph so as to arrive at a schedule for instructions in the output program. (see col. 10, lines 5-9).

The Office Action states that "Whatever can be executed in software can be implemented in hardware." However, at best, this argument supports the notion that Subramanian would suggest hardware that generates an output program from a dependency graph by mapping instructions in the graph according to associated physical times. It would not suggest hardware that stores loop parameters in control logic that is used to cause the instructions to be selectively issued to functional units, as explicitly required claims 35 and 43.

Moreover, Subramanian teaches nothing about storing a kernel set of instructions in a processor, and associated loop parameters, and then selectively issuing the stored **kernel** set of instructions using the loop parameters "so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, **the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions.**" Nowhere does Subramanian teach or suggest such a technique. Rather, Subramanian requires explicitly defining the prologue and epilogue sets of instructions separately from the kernel set of instructions. Accordingly, the alleged combination would not have met all the limitations of independent claims 35 and 43.

For at least the foregoing reasons, claims 35 and 43, together with claims 36-42 and 44-50 that respectively depend therefrom, patentably define over Mahilingaiah and Subramanian and the § 103 rejection thereof should be withdrawn.

Claims 41 and 49 Further Patentably Define Over Mahalingaiah and Subramanian

Similar to claims 15 and 16 discussed above, claims 41 and 49 further require that the control logic is operative so that the **fetch unit can be shut down** during execution of a number of iterations of the loop body corresponding to the stored plurality of instructions.

As also discussed above, Mahalingaiah explicitly discloses a fetch unit 70 (see Figure 4). This is totally separate from alignment unit 18 on which the Office Action relies. Moreover, Mahilingaiah fails to disclose or suggest that fetch unit 70 is **ever** capable of being shut down, much less during MROM instruction execution. At best, Mahilingaiah suggests that if more than one MROM instruction is detected in an instruction block that has already been fetched from memory, only one per cycle from the block is forwarded from scan unit 72 (not the fetch unit) to the MROM unit. (see col. 17, lines 1-14)

For at least the foregoing reasons, claims 41 and 49 further patentably define over Mahalingaiah and Subramanian and the § 103 rejections thereof should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Mahalingaiah, Subramanian and Valluri

Claims 19-21 stand rejected under 35 USC 103(a) as allegedly being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 9, 11 and 13, above, and further in view of Valluri and Govindarajan's "Modulo-Variable Expansion Sensitive Scheduling" published in *High Performance Computing*, 1998 ("Valluri").

Claims 19-21 depend from claims 9, 11 and 13, which have been shown above to patentably define over Mahalingaiah, at least because Mahalingaiah does not say anything about prologue, epilogue and kernel sets of instructions, much less a technique for causing all types of instructions to be executed **based on** a stored **kernel** set of instructions and received loop parameters as required by claims 9, 11 and 13. The alleged combination with Subramanian and Valluri would not have cured this deficiency.

Moreover, Valluri is directed to compiler-based scheduling of code, which further teaches away from the invention, which is based in hardware. Accordingly, one skilled in the art would not be led to the hardware-based invention of claims 19-21, even if, *arguendo*, Valluri could be combined with Mahalingaiah and Subramanian.

For at least these reasons, the rejections of claims 19-21 should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Mahalingaiah, Subramanian and Mason

Claims 22-25, 34, 42 and 50 stand rejected under 35 USC 103(a) as allegedly being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 3, 8, 11 and 13, above, and further in view of U.S. Patent No. 6,418,489 to Mason et al. ("Mason").

Claims 22-25 depend ultimately from independent claim 1, claim 34 depends from independent claim 32, claim 42 depends from independent claim 35, and claim 50 depends from independent claim 43. These independent claims have been shown above to patentably define over Mahalingaiah and Subramanian. The further alleged combination of these references with Mason would not overcome the shortcomings of Mahalingaiah and Subramanian as discussed above. Accordingly, claims 22-25, 34, 42 and 50 are patentable at least for the reasons claims 1, 32, 35 and 50 are patentable.

Moreover, Mason does not teach the type of loop iteration required in the rejected claims. For example, claim 34 requires the control logic to be "operative to allow interrupts to be handled at the end of a current one of the number of loop iterations [associated with a kernel], and to complete the number of loop iterations after the interrupt is handled." In contrast Mason merely discloses taking interrupts at the end of loop iterations (including epilogue). In modulo scheduled code according to the present invention, there is no natural and clean end of a kernel iteration; all the iterations are overlapped. Accordingly, Mason's interrupts would not meet the limitations explicitly required by the claims.

For the foregoing reasons, claims 22-25, 34, 42 and 50 patentably define over the cited prior art and the § 103 rejection of the claims should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Mahalingaiah and Fleck

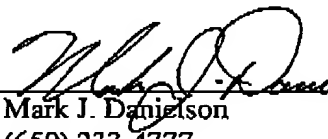
Claim 52 stands rejected under 35 USC 103(a) as allegedly being unpatentable over Mahalingaiah in view of Fleck. Although identified at paragraph 40 as claim 53, it appears that this rejection was intended for claim 52, which has been canceled herewith. Accordingly, it is rendered moot. The apparent actual rejection of claim 53 was addressed above.

Conclusion

If any issues remain which the Examiner feels may be resolved through a telephone interview, she is kindly requested to contact the undersigned at the telephone number listed below.

Respectfully submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

Date: May 1, 2006



40,580
Mark J. Danielson Reg. No.
(650) 233-4777
Please reply to customer no. 27,498